THE BRS-IMPROVED A* ALGORITHM-BASED PATH PLANNING FOR DIGITAL TWIN OF COLLABORATIVE ROBOTS

DOAN THANH XUAN[1]*, NGUYEN THANH HUNG[1], VU TOAN THANG[1]

[1]School of Mechanical Engineering, Hanoi University of Science and Technology, Hanoi 100000, Vietnam

DOI: 10.17973/MMSJ.2025_09_2024123

xuan.doanthanh@hust.edu.vn

Collaborative robots play an important role in production, assembly, operating systems, and can be used in many automation tasks. Due to the repetitive nature of robot actions, the overall efficiency of the production system is mainly determined by the precision of the robot's movement. This study proposes the optimization of robotic path planning relied on its digital twin combined with the A* algorithm. Unity is used to create the digital twin of the UR3 robot, where its path planning, based on an improved A* algorithm, is trained in a virtual environment before being implemented in the physical world. Some limitations associated with the traditional A* algorithm include redundant points, jagged paths and proximity to obstacles, increasing collision risks. Therefore, we developed the BRS-improved A* algorithm that incorporates buffer distance, redundant point elimination, and smoothing optimization using Bezier curves. Realistic movements for the robot are planned by simulated training in a virtual environment. In addition, the virtual robot's available path will be flexibly adjusted based on the physical robot's motion data, allowing for the re-adjustment of the physical robot's motion trajectory. This work uses a robot path planning algorithm to optimize the robot's path by reducing errors in the physical robotic path through interaction between virtual and real data. Additionally, optimizing the robot's path reduces the distance it needs to travel, thereby increasing energy efficiency for both the robot and the entire system.

KEYWORDS

Improved A* algorithm, digital twin, UR3 robot, path planning

1 INTRODUCTION

Collaborative robots are extensively utilized in production systems, where humans and robots work together, rapidly taking on essential roles in daily life. These robots have to plan their movements to align with the tasks they perform within the system, ensuring minimal collisions with humans. Over the last decades, a collision-free route or path planning for a moving entity, such as a robot, within a given environment has been subject to numerous studies [Cai 2019, Costa 2019, Madava 2019, Ab Wahab 2020, Wang 2020]. Classic path planning algorithms encompass the Ant Colony Optimization (ACO) [Kumar 2018, Ning 2018, Maheshwari 2021], Genetic Algorithm (GA) [Elhoseny 2018, Nazarahari 2019], Rapidly-exploring Random Tree (RRT) [Hauer and Tsiostras 2017, Hirakawa 2019], and the A* algorithm [Hart 1968, Liu 2019, Sedighi 2019]. The A* algorithm is particularly popular among these methods, recognized for its graph search approach that effectively finds the optimal path through iterative refinement. The effectiveness of the A* algorithm is mainly shown through the speed of path planning and the reliability of the chosen route. Despite of being extensively studied, the application of this algorithm still suffers from some flaws, such as redundant points and jagged paths. These factors reduce the certainty of the planned route. Therefore, the performance of the algorithm and the robot's speed are being evaluated by examining the speed of path planning and the smoothness of the path. To strengthen the traditional A* algorithm, the proposed approach should include measures to eliminate redundant points and create a smoother path. This research paper aims at proposing three methods to improve the traditional A* algorithm, which include buffer distance, redundant point elimination, and smoothing. Simulation experiments were conducted to evaluate the efficiency of the traditional A* algorithm compared to the BRS-improved A* algorithm in path planning. The BRS-improved A* algorithm was also proven to be effective when it was integrated into the UR3 collaborative robot hardware platform and tested in real-world scenarios.

Digital twins, recognized as a pivotal technology for realizing the concept and objectives of smart manufacturing, have garnered significant attention from academia and are increasingly being deployed across various industrial sectors [Jiang 2021, Hao 2022]. By enabling information interaction between physical and virtual environments, they digitalize models of physical objects for simulating these objects' operations in the physical environment and autonomously optimizing the operational states of these objects [Fei 2019, Lu 2020, Jiang 2021]. As a technology that fully utilizes models, data, and integrates multiple fields, digital twins aim at the entire product lifecycle process, acting as a bridge linking the physical and digital realms to provide more intelligent, efficient, and real-time services [Jiang 2019, Liao 2021]. Accurate distribution based on digital twins for manufacturing operations is an application that combines digital twin technology and robots [Li 2019, Li 2020].

In this paper, the combination of BRS-improved A* algorithm and digital twin was proposed in collaborative robot path planning. The novelty of this study lies in integrating a digital twin of the UR3 robot with a BRS-improved A algorithm* (Buffer distance, Redundant point elimination, Smoothing) for collaborative robot path planning. This is the first time this combination has been proposed to enhance robot motion precision, path safety, and real-time optimization through virtual–real interaction.

2 BRS-IMPROVED A* ALGORITHM

2.1 *Basic theory of the traditional A* algorithm*

The A* algorithm is one of the most well-known path planning algorithms; it was first presented and thoroughly explained in [Hart 1968]. It is designed as a heuristic search algorithm that looks for the least expensive path between the initial and goal nodes by examining every option. This algorithm stands out from other blind search algorithms by utilizing heuristic information relevant to the problem's characteristics to direct its search effectively [Fu 2018]. It works by estimating the distance on a 2D plane between any given node and the goal node using an Open list, a Closed list, and a heuristic function.
The reason A* is regarded as a best-first algorithm is that it evaluates each cell within the configuration space based on the value of:

$$f(n)=g(n)+h(n) \quad (1)$$

where: $g(n)$ represents the total cost incurred along the path from the start node to the current node, n. $h(n)$ denotes the

estimated cost from the current node n to the goal node, calculated using the heuristic function. $f(n)$ is the evaluation function for node n.

The node with the smallest $f(n)$ is chosen as the next node in the path after the algorithm accesses each neighbouring node of the currently evaluated node using $f(n)$. The A* algorithm has the benefit of allowing for the adoption, modification or addition of different distance measures as standard distances.

2.2 *Proposed method*

Buffer distance

The path produced by the traditional A* algorithm may pass near obstacles. A collaborative robot that follows this path, has a significant danger of colliding with obstacles. Consequently, it is crucial to keep a suitable distance from obstacles when planning a path. In this study, the concept known as "buffer distance" was applied [Wang 2022] i.e., leaving extra room around obstacles as safety precaution. Robots design their routes using rasterized maps, and the buffer distance extends beyond obstacles by using a grid as its fundamental unit. It is a function of the robot's speed, the size of the robot model, and the number of grids, indicating the closest distance the path can approach obstacles.

Nodes within the buffer distance, along with the obstacles, will not be visited during path planning. This buffer distance acts as a "collision buffer" between the robot and obstacles, significantly reducing the risk of collisions during movement. Thus, buffer distance enhances both the algorithm's robustness and efficiency. It reduces effectively the map scale, since the algorithm avoids visiting the extended nodes. As a result, the algorithm's overall traversal of nodes is lowered, increasing its efficiency. Fig. 1 displays the schematic diagram of buffer distance.

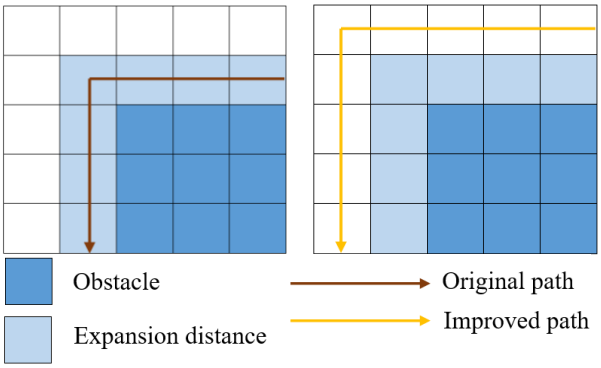

Figure 1. Schematic diagram of buffer distance

When it comes to choosing the buffer distance size, the rasterized map's grid is usually selected by default. We chose one grid to be the buffer distance in the simulation tests. The buffer distance typically defaults to the radius of the cylinder or sphere in real-world settings. This distance minimizes the amount of journey space wasted while ensuring the path's dependability. The bilateral buffer distances equal the robot's size when there are impediments on both sides.

It is important to consider how the automatic calculation of buffer distance is managed for various environments. To determine the buffer distance, the robot equivalent model was used and the following assumptions were made to simplify the model.

The robot model has a radius of r and a cruising speed Vr, which must adhere to the condition $V_r \le V_{max}$ with V_{max} being the maximum cruising speed based on the robot's performance. V_r is a speed threshold, and V_i is the current speed of the robot. When $V_i \le V_r$, the buffer distance is extended by only one node. Nevertheless, the chance of a collision between the robot and obstacles rises when the present speed surpasses V_r, requiring an increase in the number of enlarged nodes. One or more square grids can be used to represent the obstacle. According to the mapping rule between the robot model and the map, the robot's radius corresponds to the length of one grid unit. The relationship between the current speed and the speed threshold determines the connection between the number of expansion nodes and the robot's speed.

$$E(V_i) = \begin{cases} r & V_i \le V \\ \dfrac{V_i}{V_r} r & V_r < V_i \le V_{max} \end{cases} \quad (2)$$

where the number of expansion nodes is represented by $E(V_i)$. In terms of calculating the magnitude of the buffer distance, the cylinder's radius is used as the default value when $V_i \le V_r$. This distance offers a sufficient "collision buffer" to ensure the path's reliability while minimizing wastage of physical space the robot travels through. When obstacles are present on both sides, bilateral buffer distances are equivalent to the robot's own dimensions. As the robot's speed increases, the risk of collisions also increases. Therefore, expanding the buffer distance ensures the algorithm's robustness remains intact. $E(V_i)$ should increase as V_i increases; otherwise, the risk of robot collisions rises. Thus, the calculation of the buffer distance exhibits a linear correlation with the speed V_i.

Redundant point elimination

The redundant point elimination method initially improves by pre-planning a local path from the current node to the goal node and subsequently exploring the surrounding area of the current node. If this local path is deemed safe and collision-free, it is directly traversed. Additionally, the method employs post-processing techniques to optimize the resulting path, particularly by straightening the local path, thereby reducing the number and length of required local paths.

In this method, the probabilistic motion planning's query phase utilizes an improved version of the A* algorithm [Sánchez 2002, Clark 2005]. The probabilistic motion planning consists of two stages: preprocessing and querying. During preprocessing, collision-free sample points are randomly generated within the robot's workspace, which serve as nodes in the subsequent stages.

Subsequently, local path planning constructs safe and collision-free paths between these sample points. These paths are validated for feasibility and collision avoidance by mapping them into the robot's configuration space, considering shared constraints such as velocity, acceleration, and energy optimization. Thus, collision-free sample points and safe local paths form the components of the probabilistic roadmap.

In the querying stage, employing the redundant point elimination method, probabilistic motion planning generates a path that searches for and obtains a safe path for the robot's movement from the initial node (Starting point) to the goal node (Ending point).

In the redundant point elimination method, the waypoints identified by the traditional A* algorithm as the robot's path undergo a calculation. The method evaluates local connections to subsequent waypoints to ascertain whether they pass through obstacles. If a connection avoids intersecting with obstacles, the intermediate waypoint is bypassed. This method significantly decreases the number of waypoints the robot must traverse, while still guaranteeing a valid path from the starting

point to the ending point as specified in the original problem statement. The flowchart illustrating the redundant point elimination method (Fig. 2) is outlined as follows [Fu 2018]:

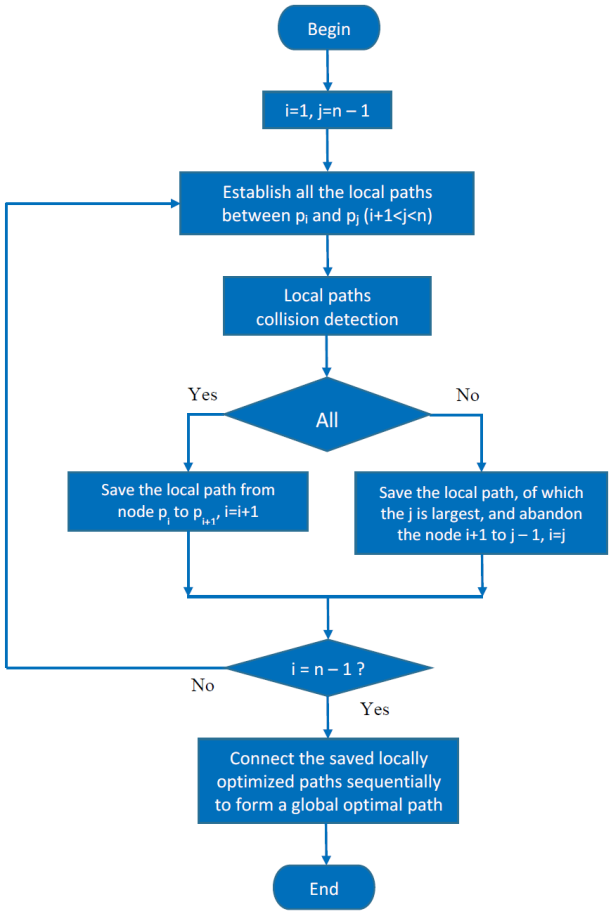

Figure 2. The flowchart of the improved A* algorithm (redundant point elimination)

Step 1: Start by initializing i to 1 and j to n − 1, where i and j represent indices of the current nodes being evaluated along the robot's path stored in the CLOSED list. Optimize the path from the initial node p_1 to the goal node p_n.

Step 2: Define all local segments between node p_i and p_j, where j ranges from i + 1 to n − 1.

Step 3: Verify if the local segments encounter obstacles. If all local paths result in collisions, proceed to Step 4. Otherwise, move to Step 5.

Step 4: Record the local segment between node p_i and p_{i+1}, then increment i by 1 (i = i + 1).

Step 5: Store the local segment containing node p_j with the highest j value, replace i + 1 with j − 1, and set i = j.

Step 6: Check if i equals n − 1 (indicating that node i is close to the last node). If yes, proceed to Step 7. If not, return to Step 2.

Step 7: Connect the nodes in the locally optimized path that was previously stored, resulting in a sequential connection of the globally optimized path.

The optimization results achieved by the improved A* algorithm (redundant point elimination) compared to the traditional A* algorithm are illustrated in Figs. 3-6.

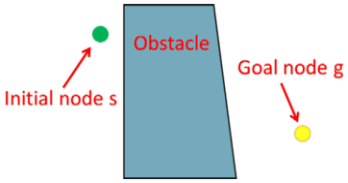

Figure 3. A simple example of a robot task

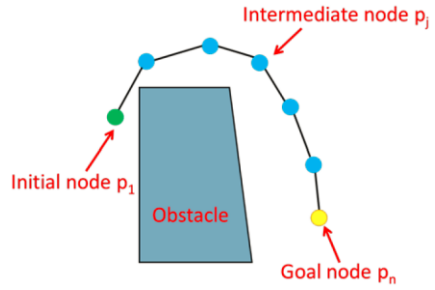

Figure 4. A* algorithm-based path planning (traditional)

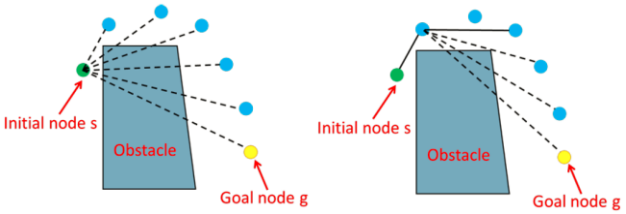

Figure 5. Optimal result of the first step on the left, the second step on the right

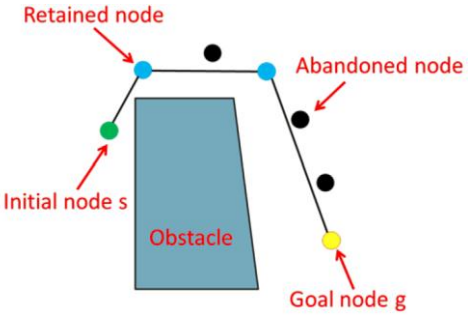

Figure 6. The optimal result of the final step

In summary, the redundant point elimination method plays a vital role in optimizing the planned trajectory by evaluating and eliminating unnecessary intermediate waypoints when a direct connection between two non-consecutive nodes is both feasible and collision-free. This approach significantly reduces the total number of path points, thereby shortening the robot's travel path and improving its operational efficiency. Moreover, it maintains the validity of the path while concurrently lowering the computational burden and execution complexity associated with robot navigation.

Smoothing optimization

Paths can be smoothed with the use of Bezier curves. A space curve with advantageous geometric qualities is known as a Bezier curve, and it was first proposed by French engineer Pierre Bezier in 1962. Often used in computer graphics and computer-aided design, it does not always go through all of its control points, or defining data points. When these points form a convex polygon, the resulting Bezier curve is also convex, distinguishing it from other curves like cubic splines or polynomials. This curvature has fewer turning points, enhancing its smoothness.

Key properties of Bezier curves include:

1. Symmetry: Each coefficient of the curve is symmetrical with its reciprocal counterpart.
2. Convex Hull: The curve always resides within the convex hull defined by its control points.
3. Endpoint Consistency: The first and last control points directly correspond to the starting and ending points of the Bezier curve.
4. Recursion: The coefficients of the Bezier curve adhere to a recursive formula as follows:

$$B_{i,n}(t) = (1-t)B_{i,n-1}(t) + tB_{i-1,n-1}(t), \{i = 0,1,...,n\} \quad (3)$$

Because a Bezier curve's higher-order derivatives are continuous, its curvature smoothly changes from the starting point to the ending point. Using degree *n* Bernstein basis polynomials, a parametric curve is defined as a Bezier curve of degree n:

$$P(t) = \sum_{i=1}^{n} p_i B_{i,n}(t), t \in [0,1] \quad (4)$$

Here, *t* denotes the normalized parameter, $P_i(x_i,y_i)^T$ represents the coordinate vector of the i^{th} control point, $B_{i,n}$ denotes the Bernstein basis polynomials, which serve as the basis functions in the Bezier curve expression:

$$B_{i,n}(t) = C_n^i t^i = \frac{n!}{i!(n-i)!} t^i (1-t)^{n-i}, i = 0,1,...,n \quad (5)$$

The control points have an impact on the Bezier curve's derivatives. Equation 6 provides the first derivative of the Bezier curve, which is obtained from Equation 4.

Furthermore, the computation of higher-order derivatives of the Bezier curve is also possible.

$$\dot{P}(t) = \frac{dP(t)}{dt} = n\sum_{i=0}^{n-1} B_{i,n-1}(t)(P_{i+1} - P_i) \quad (6)$$

The Bezier curve's curvature with respect to t in two dimensions is represented as:

$$k(t) = \frac{1}{R(t)} = \frac{\dot{P}_x(t)\ddot{P}_y(t) - \dot{P}_y(t)\ddot{P}_x(t)}{(\dot{P}_x^2(t) + \dot{P}_y^2(t))^{1.5}} \quad (7)$$

In the context of path planning for robots, Bezier curves are concatenated to create smooth paths, as depicted in Fig. 7.

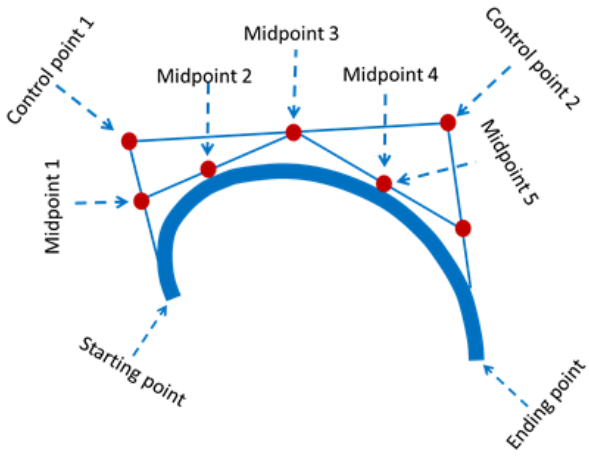

Figure 7. Smoothing the path using Bezier curve technique

Thus, the incorporation of the Bezier curve into the path planning process effectively smooths the robot's trajectory by replacing sharp corners and jagged segments with continuous, gentle curves. This technique enhances path continuity and significantly reduces the number of abrupt turning points. As a result, it improves the stability of the robot's motion and allows for smoother and faster navigation by minimizing sudden directional changes.

2.3 Analysis of the BRS-Improved A Algorithm's Time Complexity*

In this section, the flowchart of the BRS-improved A* algorithm is presented in Figure 8, incorporating buffer distance, redundant point elimination, and smoothing optimization.

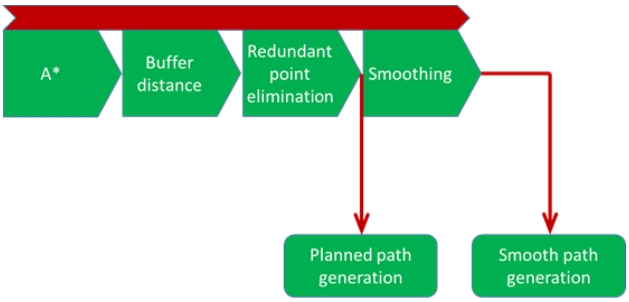

Figure 8. Execution Process of the BRS-Improved A* Algorithm

The method employs a double loop: the inner loop identifies and adds the lowest-cost point to the open list by exploring neighboring points in four distinct directions. Until the queue traversal is finished, the points in the open list are processed by the outer loop. The scale of the map, the locations of the beginning and ending points, and the characteristics of the obstacles are some of the variables that influence the time complexity of various path planning algorithms.

This algorithm is an extension of the traditional A*, incorporating three techniques: Buffer Distance, Redundant Point Elimination, and Path Smoothing using Bezier Curves. These steps are executed sequentially, so the overall computational complexity is the sum of the individual component steps:

Computational complexity of traditional A*:
$\mathcal{O}(n \log n)$

Computational complexity of Buffer Distance:
$\mathcal{O}(n)$

Computational complexity of Redundant Point Elimination:
$\mathcal{O}(d)$

Computational complexity of Path Smoothing (Bezier Curve):
$\mathcal{O}(d)$

Where:

n: the number of points in the map (for an N × N grid, n = N^2)

d: the length (number of points) in the computed path

The overall computational complexity of the BRS-Improved A* algorithm:

$$\mathcal{O}(n \log n) + \mathcal{O}(n) + \mathcal{O}(d) + \mathcal{O}(d) \approx \mathcal{O}(n \log n)$$

3 SIMULATION TESTING

3.1 Experiments and Results

For experimental part, we primarily simulate and evaluate four algorithms, including the traditional A* algorithm, buffer distance, redundant point elimination, and smoothing optimization in Python software.

Figures 9 – 12 display the path planning results after testing the four algorithms, which correspond to the maps of sizes 60×60, 120×120, 180×180, and 240×240. It can be seen that the randomly generated obstacles are indicated by the black blocks, the buffer distances are shown by the gray blocks that surround these obstacles. The starting point is represented by the orange dot, the ending point is shown by the mint blue dot, and the final path generated by the improved A* algorithm is shown by the purple line.

The overall distance and point count are shown in Tabs. 1 through 4, which show how long the intended path is. The path's resilience and smoothness are reflected by the number of right-angle turns and the maximum turning angle. Tabs. 1 through 4 show the statistical outcomes for randomized maps, which correlate to the 60×60, 120×120, 180×180, and 240×240 map sizes.

The experiment was programmed in Python software, three techniques were performed sequentially: buffer distance,

redundant point elimination, and smoothing optimization. Obstacles were arranged randomly, with a size of 6×6. The experimental results are presented in Figs. 9 – 12 and Tabs. 1 – 4.

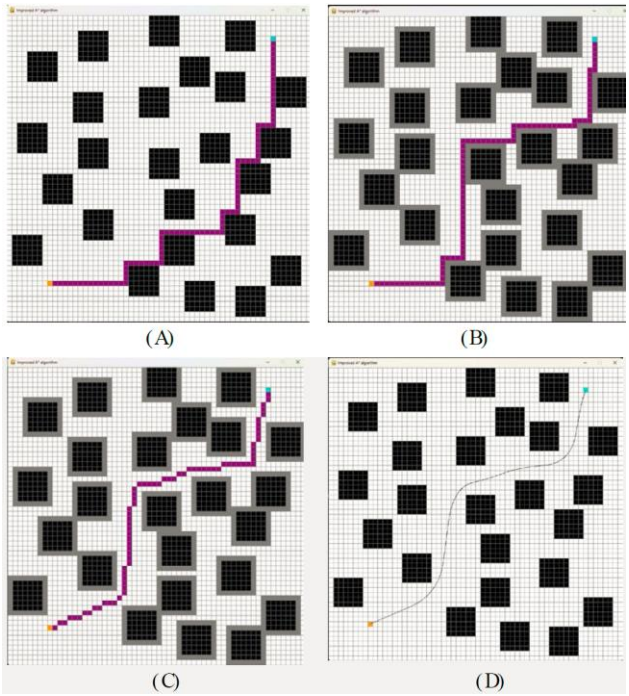

Figure 9. Simulation results of four algorithms on a 60×60 map. (A) Traditional A* algorithm. (B) A* with buffer distance. (C) A* with buffer distance and redundant point elimination. (D) BRS-improved A* algorithm

Table 1. Simulation results of four algorithms on a 60×60 map

Indicators	Traditional A* algorithm	A* with Buffer distance	A* with Buffer distance and Redundant point elimination	BRS-improved A* algorithm
Number of points	93	93	76	76
Number of right-angle turns	11	11	0	0
Max turning angle	90	90	45	-

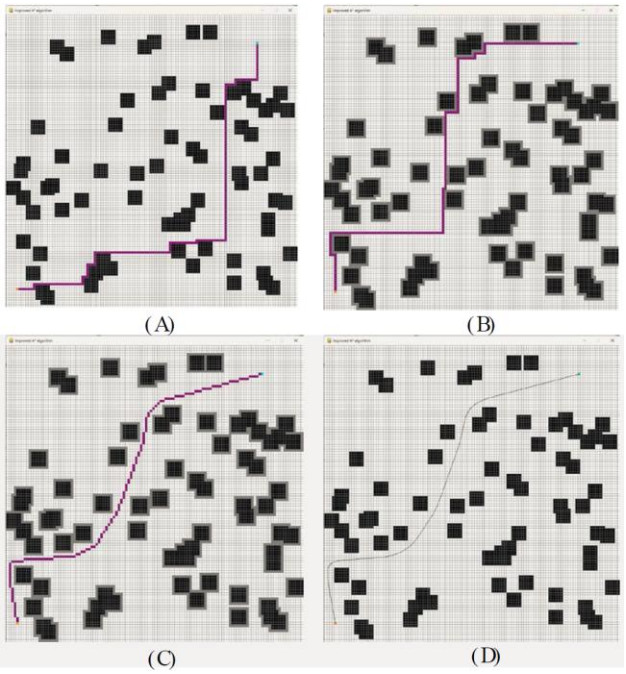

Figure 10. Simulation results of four algorithms on a 120×120 map. (A) Traditional A* algorithm. (B) A* with buffer distance. (C) A* algorithm with buffer distance and redundant point elimination. (D) BRS-improved A* algorithm

Table 2. Simulation results of four algorithms on a 120×120 map

Indicators	Traditional A* algorithm	A* with Buffer distance	A* with Buffer distance and Redundant point elimination	BRS-improved A* algorithm
Number of points	201	205	159	161
Number of right-angle turns	17	13	0	0
Max turning angle	90	90	45	-

(2)

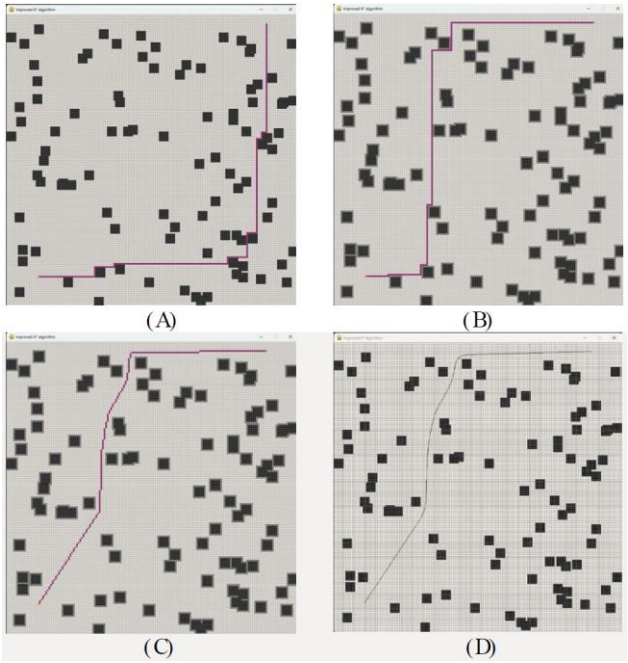

Figure 11. Simulation results of four algorithms on a 180×180 map. (A) Traditional A* algorithm. (B) A* with buffer distance. (C) A* algorithm with buffer distance and redundant point elimination (D) BRS-improved A* algorithm

Table 3. Simulation results of four algorithms on a 180×180 map

Indicators	Traditional A* algorithm	A* with Buffer distance	A* with Buffer distance and Redundant point elimination	BRS-improved A* algorithm
Number of points	298	298	239	235
Number of right-angle turns	13	10	0	0
Max turning angle	90	90	45	-

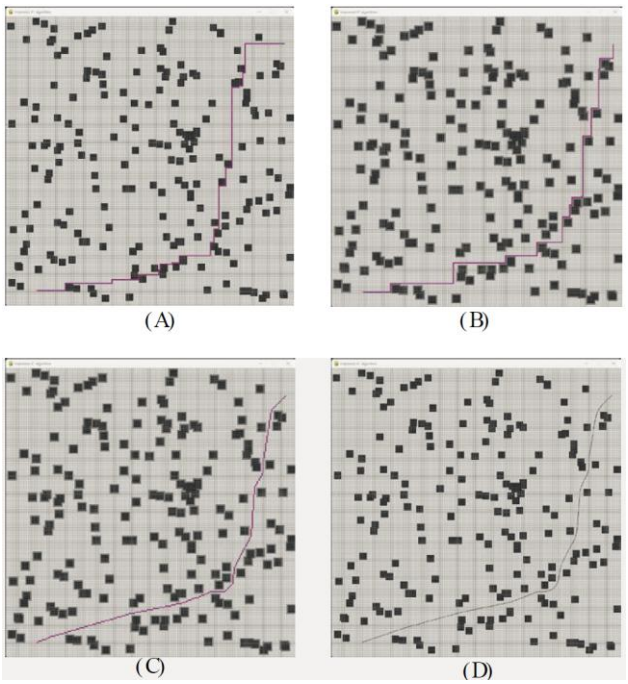

Figure 12. Simulation results of four algorithms on a 240×240 map. (A) Traditional A* algorithm. (B) A* with buffer distance. (C) A* algorithm with buffer distance and redundant point elimination (D) BRS-improved A* algorithm

Table 4. Simulation results of four algorithms on a 240×240 map

Indicators	Traditional A* algorithm	A* with Buffer distance	A* with Buffer distance and Redundant point elimination	BRS-improved A* algorithm
Number of points	413	413	320	322
Number of right-angle turns	30	25	0	0
Max turning angle	90	90	45	-

3.2 Analysis of experimental results

As shown in Tab. 1, the number of path points after applying the BRS-improved A* algorithm (the final column of the table) is reduced from 93 points to 76 points when compared to the traditional A* algorithm, corresponding to an 18.3% reduction. After applying redundant point elimination, all right-angle turns are smoothed, resulting in no right-angle corners and reducing the maximum turning angle to 45˚.

The data in Tabs. 2-4 indicate that the effectiveness of reducing the number of path points increases as the grid resolution of the map increases. Specifically, the results in Tabs. 2-4 correspond to maps with resolutions of 120×120, 180×180 and 240x240, with the respective reduction rates of the number of path points being 19.9, 21.1, and 22%.

A general overview of the results for all four maps shows that the buffer distance only increases the buffer between the moving object and the obstacles to reduce the likelihood of a collision, without significantly reducing the number of points traversed compared to the traditional A* algorithm. It can be observed that the number of points in Column 2 (A* with Buffer distance) does not differ much from Column 1 (Traditional A* algorithm). When Redundant Point Elimination was applied, it significantly reduced the number of points traversed. For example, in Tab. 2, the number of points decreased from 205 in Column 2 (A* with Buffer distance) to 159 in Column 3 (A* with Buffer distance and Redundant Point Elimination), reducing the number of right angles to zero. Similar data can be seen in the remaining tables. The Smoothing Optimization technique did not significantly change the number of points but helped smooth the path for the moving object (as seen in Figure D on the maps).

Compared with the traditional A* algorithm, the BRS-improved A* algorithm, which integrates three techniques — buffer distance, redundant point elimination, and smoothing optimization — combines the advantages of all these techniques to reduce path points, minimize the number of turns, and enhance path smoothness. By utilizing buffer distances in the improved A* algorithm, padding points are incorporated to create barriers, which effectively prevent collisions and enhance the robustness of the path.

4 REAL-WORLD CASE

This section describes the application of the improved A* algorithm presented above in conjunction with the digital twin

method to optimize the UR3 robot's path. Fig. 13 illustrates the interactive functionality between the virtual model of the robot and its physical counterpart within the digital twin framework. The robot's virtual model is developed in Unity, enabling real-time bidirectional data exchange through Unity's virtual serial port between the physical UR3 robot and the virtual representation.

4.1 Digital twin of the robot

Physical robot

The robot used in the experiment is the UR3, a six-degree-of-freedom robot. The first three joints are revolute, functioning similarly to a human arm and responsible for positioning, while the last three joints are spherical, serving as orientation joints for the robot's end-effector. This six-degree-of-freedom configuration provides the robot with sufficient flexibility and complexity for both positioning and orientation tasks. Although there is a wide variety of robot types and quantities, the six-degree-of-freedom structure is common in industrial robots. The UR3 collaborative robot, as detailed on the Universal Robots website [Universal Robots 2024], weighs only 24.3 lbs (11 kg) but has a payload capacity of up to 6.6 lbs (3 kg). It can rotate ±360 degrees on all wrist joints and has unlimited rotation on the final joint.

The UR3 robot is equipped with a force sensor to limit the collision force with humans or obstacles. The default force threshold is 150 N, with a minimum configurable limit of 50 N. In the event of a collision, the force sensor enables the robot to immediately stop, thereby ensuring operational safety.

An independent camera system is employed to capture 3D images of humans and obstacles within the robot's workspace. The acquired data are used to construct obstacle models in the software, which serve as input for the robot's path planning and obstacle avoidance algorithms.

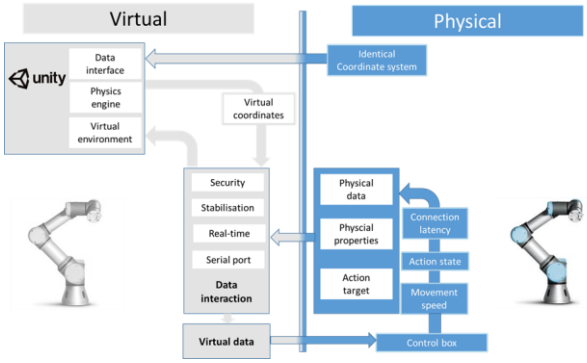

Figure 13. Interaction framework between virtual models and physical entities

In the actual experiment, the UR3 robot will find a path from the starting point to the ending point with obstacles arranged as shown in Fig. 14.

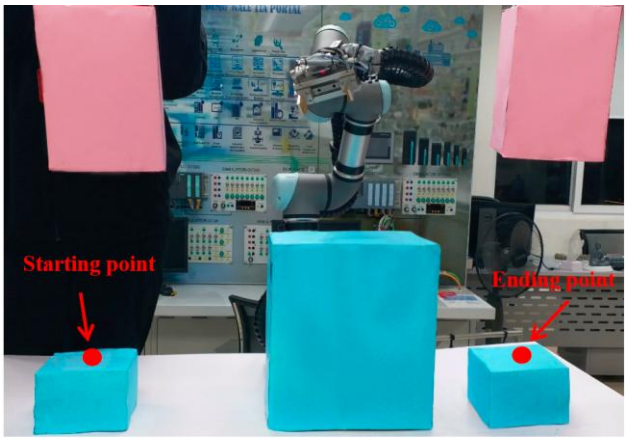

Figure 14. Images of the UR3 robot and obstacle system in real-life scenarios

Virtual environment construction

A virtual model is a virtual replica of a physical object, and consists of four layers: geometry, behavior, interaction, and association. In geometric models, the relationships (such as size and shape) are described [Xiao 2021]. In behavioral modelling, the behaviors (i.e., expected, observed, and random) are analyzed. In interaction modelling, the virtual model and physical object interact with each other in terms of data, behavior, and information [Liu 2022, Yun 2022]. Association between the geometric, behavioral, and interaction models are described by the association model.

Using Unity, we built the virtual model, controlled its motion, and created the experimental scenes as demonstrated [Parak 2024]. Fig. 15 shows a robot movement scene in Unity, which encompasses a virtual robot model and an obstacle system. Developing a virtual model necessitates taking into account realistic elements, including the robot's material, weight, and speed. The simulation generates a virtual robot that replicates the characteristics of the physical robot, also known as a digital twin. The virtual robot simulation runs continuously and in parallel with the physical robot. All state changes in the physical robot are reflected in the virtual robot, and vice versa. The simulation results of the virtual robot dynamically change according to the physical robot's operations. The digital twin serves as a critical component in this study by offering a high-fidelity virtual simulation environment, for pre-training and testing the robot's motion planning strategies. Through seamless bidirectional data exchange between the virtual model and the physical UR3 robot, the system facilitates continuous synchronization and real-time feedback for path correction. This interactive framework enables dynamic adjustment of the robot's motion trajectory, thereby enhancing operational precision, adaptability to environmental changes, and overall robustness of the robotic system. Tab. 5 provides the detailed parameters of the experimental environment. Regarding the robot's trajectory, it is essential for it to begin from the starting position, navigate through obstacles, and finally reach the ending point.

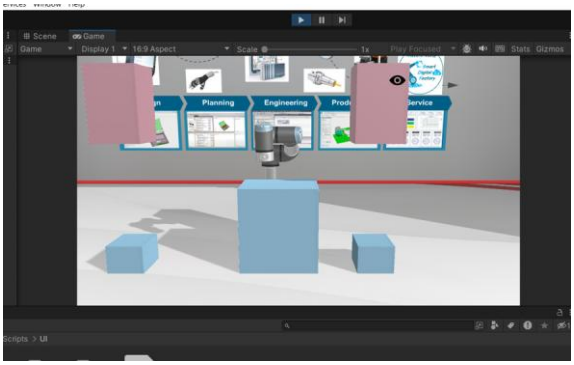

Figure 15. Images of the UR3 robot and obstacle system in the virtual environment

Table 5. Experimental environment parameters

Facility name	Dimensional parameters
Starting point coordinates	(-480; -252.5; 100)
Ending point coordinates	(290; -252.5; 100)
Size of the pink obstacle	15 cm × 17 cm × 26 cm
Size of the blue obstacle in the middle	21 cm × 28 cm × 30.5 cm
Distance from the starting point to the left plane of the blue obstacle in the middle	23 cm
Distance from the ending point to the right plane of the blue obstacle in the middle	13 cm

Experimental Procedure

The virtual model is a tool for precise mapping, simulation, and feedback refinement in digital twin technology. Several simulation tests are carried out within the virtual environment to identify suitable operating parameters, which are then transmitted to the actual system, enabling the virtual model to control it [Liu 2021]. Before starting experiments, we ensure that consistent values are set for both the movement speed of the physical robot and the lifting speed of the electric actuator. By using an orthogonal encoder for global positioning, the physical robot can navigate directionally by defining the coordinates of the ending point.

Within digital twin technology, the virtual model functions as a reflection of reality, simulating and making adjustments based on feedback received from the physical model. Following multiple assessments of the system's performance within the virtual environment, suitable operating parameters are identified and transmitted to the physical system to facilitate control of the virtual model via the physical model [Liu 2021]. Before beginning the experiment, we set a movement speed for the physical robot, the starting and ending positions of the robot during its movement, and a predefined system of obstacles.

There are three data classes exchanged between the physical robot and its digital twin, including:

• The “ur_stream_data” class is used to transmit information from the physical robot to the virtual robot, including: connection port address, joint angles, end-effector coordinates, and end-effector orientation angles.

• The “ur_control_data” class is used to send control signals from the Unity simulation software to the physical robot, including: connection port address, command byte sent to the robot, and the state of the 12 virtual control buttons in the software interface.

• The “global_variable_main_control” class is used to store connection status variables.

These data are exchanged in a continuous stream from the moment the connection is established until a disconnect signal is received, with a time step of 8 milliseconds between each data exchange cycle.

The specific operational procedure is as follows:

A common coordinate system is chosen for the robot's physical system and the virtual replicate. After the path coordinates are received, the physical robot receives the coordinates of the Starting and Ending points so it can move in accordance with the coordinates. The path the real robot travels is the one it learned during its training in the simulated setting. In the virtual environment, the improved A* algorithm helps the robot find the optimal path from the Starting point to the Ending point, with the result being a path for the physical robot to follow.

Subsequently, the physical robot initiates movement based on the trajectory determined within the virtual environment, considering the coordinate alignment with the actual surroundings, and adjustments are made as necessary. At the same time, the real-time coordinate data of the path is transmitted to the virtual environment. Using the incoming real-time coordinates, the virtual model moves inside the virtual environment and detects any deviations in the actual robot's track from the predefined path. Through the use of digital twin technology, the coordinates of the path's important locations are constantly modified in accordance with the path deviation, thus increasing the precision of the actual motion trajectory.

4.2 Results

The improved A* algorithm is used in the virtual environment to find the path for the robot. The algorithm is applied in a sequence of three steps: buffer distance to help reduce the possibility of collisions when attaching end tools to the robot, such as a gripper tool to move objects from Starting point to Ending point; reducing intermediate points to eliminate redundant points; and decreasing the robot's travel time. For example, in this experiment, the number of points was reduced by using the improved A* algorithm instead of the traditional A* algorithm i.e., reducing from 4 points to 2 intermediate points; the robot's path length decreased from 1470 mm to 1125 mm, corresponding to a reduction of 23.5%. As the path complexity and the number of obstacles increase, the efficiency also improves.

Figs. 16-18 show the robot's path when the improved A* algorithm is applied step by step.

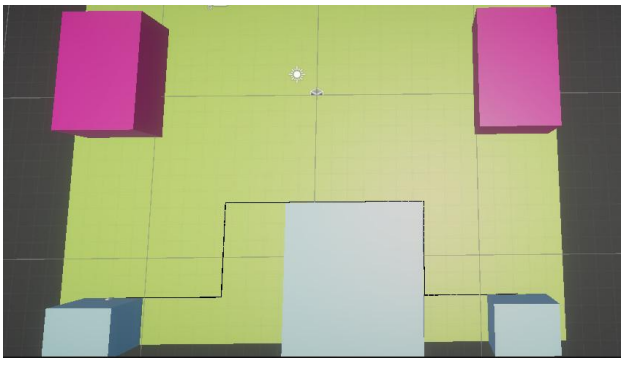

Figure 16. Robot path found using the traditional A* algorithm

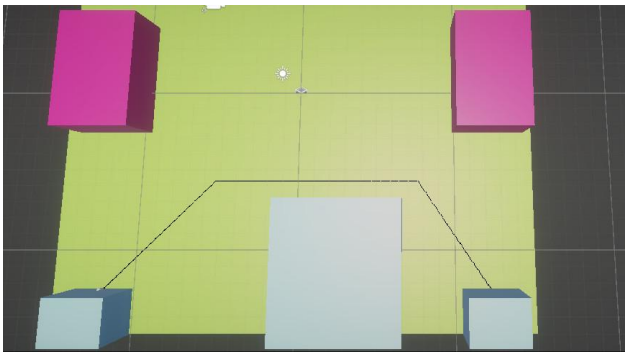

Figure 17. Robot path after applying buffer distance and redundant point elimination

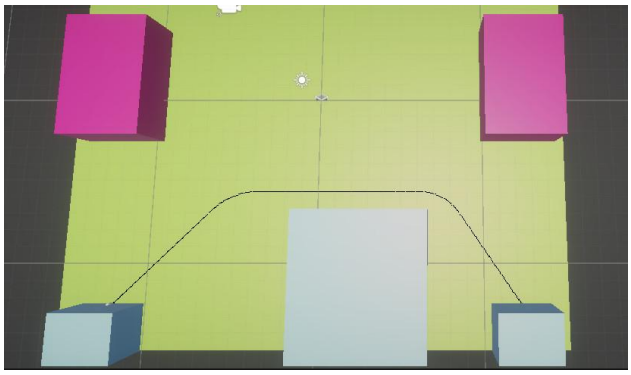

Figure 18. Robot path after applying smoothing step

Fig. 19 shows the robot's path in the virtual environment found after applying the BRS-improved A* algorithm.

Figure 19. The BRS-improved A* algorithm with 3 steps applied to the virtual robot in Unity

Then, the real robot with the IP address 192.168.1.6 is connected to its digital twin robot as shown in Fig. 20.

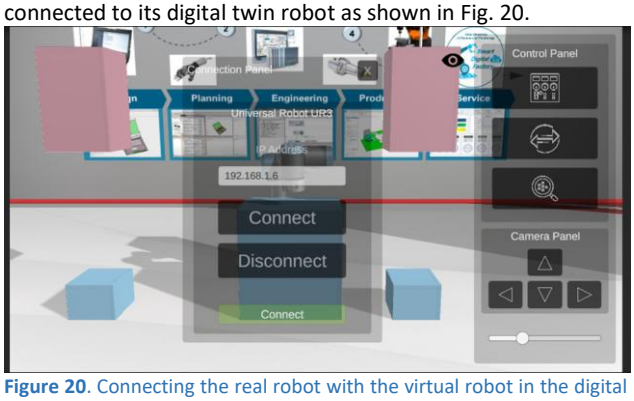

Figure 20. Connecting the real robot with the virtual robot in the digital twin

The resulting path found in the virtual model is applied to the physical model to verify its alignment with reality, making any necessary adjustments before being implemented in the actual system.

5 DISCUSSION AND CONCLUSION

In this paper, the BRS-improved A* algorithm has been studied, which includes three techniques: buffer distance, redundant point elimination, and smoothing optimization. Simulation test results indicate that the reduction in the number of movement points achieved 18.3, 19.9, 21.1, and 22% corresponding to 60×60, 120×120, 180×180 and 240×240 maps, respectively. Additionally, the number of right-angle turns was reduced to zero, resulting in smoother and more efficient paths. Then, a specific study on a device is conducted, specifically an experiment with the UR3 robot applying the digital twin combined with the BRS-improved A* algorithm to find a path that avoids obstacles.

A virtual model that mirrors a physical entity is developed in Unity, creating a virtual environment that reflects real-world conditions. Utilizing the interactive data features of digital twin technology, the robot's route in the virtual setting is crafted through the BRS-improved A* algorithm. This path is then dynamically modified in the physical system to ensure alignment with actual conditions, enhancing the accuracy of the robot's movements and facilitating the integration of virtual and real closed-loop control. The primary focus of this research is on the establishment of the UR3 robot's digital twin, ensuring seamless communication between the virtual model and the physical counterpart, and optimizing the motion trajectory of the UR3 robot. The path length of the UR3 robot was reduced by 23.5% when applying the BRS-improved A* algorithm compared to the traditional A* algorithm.

The combination of digital twins and the BRS-improved A* algorithm has the following advantages. Digital twins create a dynamic virtual model of the physical environment, continuously updated with real-time data from sensors and other sources. This up-to-date model allows the BRS-improved A* algorithm to calculate the most precise and efficient paths by accounting for current conditions and obstacles, resulting in improved accuracy and adaptability in dynamic environments [Denk 2022]. Digital twins enable the simulation of complex and multi-layered environments, both in 2D and 3D. This capability is crucial for industries like manufacturing and logistics, where robots must navigate intricate layouts. The combination with the BRS-improved A* allows for efficient path planning that considers multiple layers of information, optimizing routes in real-time. By providing a detailed and accurate virtual representation of the physical system, digital twins help in predictive maintenance. Potential issues can be identified and addressed before they cause significant downtime, reducing maintenance costs and improving overall operational efficiency.

The response of the robot, within the framework of the proposed BRS-Improved A* algorithm integrated with the digital twin, refers to the robot's capability to accurately execute the optimized path generated in the virtual environment and to adapt reliably to real-world operational conditions, ensuring both precision and safety during navigation.

Future research will expand to incorporate digital twins and the BRS-A* algorithm for pathfinding in multiple robots operating concurrently in more complex environments. These environments will include unexpected mobile obstacles, necessitating real-time, faster, and more precise reactions from the robots.

ACKNOWLEDGEMENT

This research is funded by Hanoi University of Science and Technology (HUST) under project number T2023-PC-010.

REFERENCES

[Ab Wahab 2020] Ab Wahab, M.N., et al. A comparative review on mobile robot path planning: Classical or meta-heuristic methods? Annual Reviews in Control, 2020, Vol. 50, pp. 233-252. DOI: 10.1016/j.arcontrol.2020.10.001

[Cai 2019] Cai, K., et al. Mobile robot path planning in dynamic environments: A survey. Instrumentation. Vol. 2019, 6, pp. 92-102.

[Clark 2005] Clark, C.M. Probabilistic road map sampling strategies for multi-robot motion planning. Robotics and Autonomous Systems, 2005, Vol. 53, pp. 244-264. DOI: 10.1016/j.robot.2005.09.002

[Costa and Silva 2019] Costa, M.M. and Silva, M.F. A survey on path planning algorithms for mobile robots. In Proceedings of the 2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), 2019, pp. 1-7.

[Denk 2022] Denk, M., et al. Generating digital twins for path-planning of autonomous robots and drones using constrained homotopic shrinking for 2D and 3D environment modeling. Applied Sciences, 2022, Vol. 13, 105. DOI: 10.3390/app13010105

[Elhoseny 2018] Elhoseny, M., et al. Bezier curve based path planning in a dynamic field using modified genetic algorithm. Journal of Computational Science, 2018, Vol. 25, pp. 339-350. DOI: 10.1016/j.jocs.2017.08.004

[Fei 2019] Fei, T., et al. Digital twin five-dimensional model and ten domain applications Computer Integrated Manufacturing System, 2019, Vol. 25, No. 1, pp. 1-18.

[Fu 2018] Fu, B., et al. An improved A* algorithm for the industrial robot path planning with high success rate and short length. Robotics and Autonomous Systems, 2018, Vol. 106, pp. 26-37. DOI: 10.1016/j.robot.2018.04.007

[Hao 2022] Hao, Z., et al. Intelligent detection of steel defects based on improved split attention networks. Frontiers in Bioengineering and Biotechnology, 2022, Vol. 9, 810876. DOI: 10.3389/fbioe.2021.810876

[Hart 1968] Hart, P.E., et al. A formal basis for the heuristic determination of minimum cost paths. IEEE transactions on Systems Science and Cybernetics, 1968, Vol. 4, No. 2, pp. 100-107.

[Hauer and Tsiostras 2017] Hauer, F. and Tsiotras, P. Deformable Rapidly-Exploring Random Trees. In Proceedings of the Robotics: Science and Systems, 2017.

[Hirakwa 2019] Hirakawa, T., et al. Scene context-aware rapidly-exploring random trees for global path planning. In Proceedings of the 2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), 2019; pp. 608-613.

[Universal Robots 2024] https://www.universal-robots.com/ (accessed on 25 August 2024).

[Jiang 2021] Jiang, D., et al. Manipulator grabbing position detection with information fusion of color image and depth image using deep learning. Journal of Ambient Intelligence and Humanized Computing, 2021, Vol. 12, pp. 10809-10822. DOI: 10.1007/s12652-020-02843-w

[Jiang 2021] Jiang, D., et al. Semantic segmentation for multiscale target based on object recognition using the improved Faster-RCNN model. Future Generation Computer Systems, 2021, Vol. 123, pp. 94-104. https://doi.org/10.1016/j.future.2021.04.019

[Jiang 2019] Jiang, D., et al. Gesture recognition based on binocular vision. Cluster Computing, 2019, Vol. 22, Suppl. 6, pp. 13261-13271. DOI: 10.1007/s10586-018-1844-5

[Kumar 2018] Kumar, P.M., et al. Ant colony optimization algorithm with internet of vehicles for intelligent traffic control system. Computer Networks, 2018, Vol. 144, pp. 154-162. DOI: 10.1016/j.comnet.2018.07.001

[Li 2019] Li, B., et al. Gesture recognition based on modified adaptive orthogonal matching pursuit algorithm. Cluster Computing, 2019 DOI: Vol. 22, pp. 503-512. DOI: 10.1007/s10586-017-1231-7

[Li 2020] Li, C., et al. Surface EMG data aggregation processing for intelligent prosthetic action recognition. Neural Computing and Applications, 2020, Vol. 32, pp. 16795-16806. DOI: 10.1007/s00521-018-3909-z

[Liao 2021] Liao, S., et al. Occlusion gesture recognition based on improved SSD. Concurrency and Computation Practice and Experience, 2021, Vol. 33, e6063. DOI: 10.1002/cpe.6063

[Liu 2019] Liu, C., et al. An improved A-star algorithm considering water current, traffic separation and berthing for vessel path planning. Applied Sciences, 2019, Vol. 9, No. 6, 1057. DOI: 10.3390/app9061057

[Liu 2021] Liu, J., et al. A digital twin-driven approach towards traceability and dynamic control for processing quality. Advanced Engineering Informatics, 2021, Vol. 50, 101395. DOI: 10.1016/j.aei.2021.101395

[Liu 2022] Liu, Y., et al. Grasping posture of humanoid manipulator based on target shape analysis and force closure. Alexandria Engineering Journal, 2022, Vol. 61, pp. 3959-3969. DOI: 10.1016/j.aej.2021.09.017

[Lu 2020] Lu, Y., et al. Digital Twin-driven smart manufacturing: Connotation, reference model, applications and research issues. Robotics and Computer-Integrated Manufacturing, 2020, Vol. 61, 101837. DOI: 10.1016/j.rcim.2019.101837

[Maheshwari 2021] Maheshwari, P., et al. Energy efficient cluster based routing protocol for WSN using butterfly optimization algorithm and ant colony optimization. Ad Hoc Networks, 2021, Vol. 110, 102317. DOI: 10.1016/j.adhoc.2020.102317

[Mandava 2017] Mandava, R.K., et al. An optimized path planning for the mobile robot using potential field method and PSO algorithm. In Proceedings of the Soft Computing for Problem Solving: SocProS 2017, Volume 2, 2019; pp. 139-150.

[Nazarahari 2019] Nazarahari, M., et al. Multi-objective multi-robot path planning in continuous environment using an enhanced genetic algorithm. Expert Systems with Applications, 2019, Vol. 115, pp. 106-120. DOI: 10.1016/j.eswa.2018.08.008

[Ning 2018] Ning, J., et al. A best-path-updating information-guided ant colony optimization algorithm. Information Sciences, 2018, Vol. 433, pp. 142-162. DOI: 10.1016/j.ins.2017.12.047

[Parak 2024] Parak, R. Unity3D Industrial Robotics - Universal Robots UR3 (Digital-Twin Application). http://www.youtube.com/watch?v=kReuJdESdz0 (accessed on 18 October 2024).

[Sánchez 2002] Sánchez, G. and Latombe, J.-C. On Delaying Collision Checking in PRM Planning: Application to Multi-Robot Coordination. The International Journal of Robotics Research, 2002, Vol. 21, No. 1, pp. 5-26. DOI: 10.1177/0278364023205564

[Sedighi 2019] Sedighi, S., et al. Guided hybrid A-star path planning algorithm for valet parking applications. In Proceedings of the 2019 5th international conference on control, automation and robotics (ICCAR), 2019; pp. 570-575.

[Wang 2020] Wang, B., et al. Mobile robot path planning in dynamic environments through globally guided reinforcement learning. IEEE Robotics and Automation Letters, 2020, Vol. 5, No. 4, pp. 6932-6939. DOI: 10.1109/LRA.2020.3026638

[Wang 2022] Wang, H., et al. The EBS-A* algorithm: An improved A* algorithm for path planning. PLoS One 2022, Vol. 17, e0263841. DOI: 10.1371/journal.pone.0263841

[Xiao 2021] Xiao, F., et al. An effective and unified method to derive the inverse kinematics formulas of general six-DOF manipulator with simple geometry. Mechanism and Machine Theory, 2021, Vol. 159, 104265. DOI: 10.1016/j.mechmachtheory.2021.104265

[Yun 2022] Yun, J., et al. Self-adjusting force/bit blending control based on quantitative factor-scale factor fuzzy-PID bit control. Alex. Eng. J. 2022, Vol. 61, pp. 4389-4397.

CONTACTS:

DOAN THANH XUAN, **Dr.**
NGUYEN THANH HUNG, Assoc. Prof. Dr.
VU TOAN THANG, Prof. Dr.
School of Mechanical Engineering, Hanoi University of Science and Technology, Hanoi 100000, Vietnam
E-mail: xuan.doanthanh@hust.edu.vn, hung.nguyenthanh@hust.edu.vn, thang.vutoan@hust.edu.vn